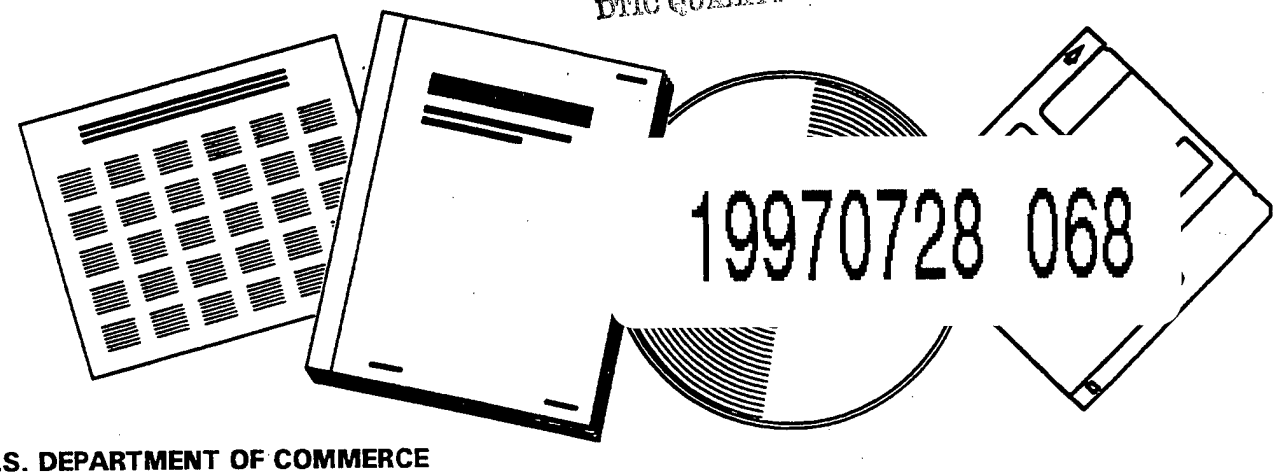
PB93-228757

COMMERCIAL APPLICATIONS OF MASSIVELY PARALLEL SUPERCOMPUTERS FOR THE 90'S

THINKING MACHINES CORP.
CAMBRIDGE, MA

1991



U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

TMC-191

Commercial Applications of Massively Parallel Supercomputers for the 90's

D.L. Waltz

Thinking Machines Corporation

Technical Report Series

TMC-191

4/91

REPORT DOCUMENTATION PAGE

PB93-228757

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1991	Technical

4. TITLE AND SUBTITLE
Commercial Applications of Massively Parallel supercomputers for the 90's

5. FUNDING NUMBERS
AFOSR
F49620-0058

6. AUTHOR(S)
D.L. Waltz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Thinking Machines Corp.
245 First Street
Cambridge, MA 02142-1264

8. PERFORMING ORGANIZATION REPORT NUMBER
TMC-191

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR -- Dept of Air FOrce
The Pentagon, Washington, DC 20330

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
"A"

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

With the dramatic increase in computing power made possible by massively parallel computers, it is now possible to dramatically speed up applications, improve their quality and accuracy, and, most importantly, speed up the software development process. A case study is provided that illustrate these points; it descrives an automatic classification system based on a massively parallel nearest-neighbor method, that was able to cut the application development time by 98% compared to an expert system solution. This application can be seen as a foundation for much more intelligent applications of the future.

14. SUBJECT TERMS
Commercial applications

15. NUMBER OF PAGES
8

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	SAR	SAR	SAR

Commercial Applications of Massively Parallel Supercomputers for the 90's[1]

David L. Waltz
Thinking Machines Corporation and Brandeis University

Abstract

With the dramatic increases in computing power made possible by massively parallel computers, it is now possible to dramatically speed up applications, improve their quality and accuracy, and, most importantly, speed up the software development process. A case study is provided that illustrate these points; it describes an automatic classification system based on a massively parallel nearest-neighbor method, that was able to cut the application development time by 98% compared to an expert system solution. This application can be seen as a foundation for much more intelligent applications of the future.

1. Introduction

Software engineering for applications -- commercial and other -- is a notoriously time-consuming and difficult process. Progress in cutting costs and increasing software productivity has been much slower than advances in hardware, so much slower that it has been argued that the software development process simply cannot be improved very rapidly. This is because programs are inherently more complex than hardware, and are inherently labor-intensive. High level languages, structured programming, CASE tools, "software factories," etc. have all made contributions, as have expert system and knowledge engineering methodologies. But it still takes years to build, test, and debug large applications, and it still seems difficult to speed up this process very much by adding more programmers -- in Brooks' words, there is no "silver bullet;" it is really the rate of progress in hardware development that is anomalous [Brooks 75, 87].

However, I argue that, for certain classes of applications (which include many commercial applications), it is possible to trade memory and MIPS/MFLOPS for knowledge engineering and software engineering. With a sufficiently large and powerful computing engine, many tasks can actually become simpler: a familiar example is that one may not have to hand-tune code to make it run sufficiently fast if hardware has excess capabilities. But I have something more radical in

Profile of the Author

Dr. Waltz directs research on artificial intelligence and development of commercial applications at Thinking Machines Corporation and is Professor of Computer Science at Brandeis University. He earned his Ph.D. from MIT in 1972, and was Professor of Electrical and Computer Engineering at the University of Illinois from 1973-1984.

[1]This research was funded in part by the Defense Advanced Research Projects Agency, administered by the U.S. Air Force Office of Scientific Research under contract #F49620-0058, and in part by the United States Bureau of the Census.

mind: sometimes an entire task becomes simpler, because a simple-to-program uniform method can produce an application with far less effort and with performance superior to those constructed using traditional methods (e.g. hand coding on small computers). The savings can, in many cases, more than justify the purchase price of sufficiently powerful hardware.

In section 2, below, I present a case study of a system which illustrates the trading of computing power and memory for programming and knowledge engineering effort. In section 3, I explain briefly the Memory-Based Reasoning paradigm which was the basis of the case study system. Section 4 explains why we should expect the MBR applications solution to work on a wide range of cases, and why the recent explosion in the numbers and sizes of databases will help.

2. Case Study: Classifying Census Returns

The U.S. Census Bureau Classification Project We recently completed an experiment with the US Bureau of the Census [Creecy, Masand, Smith & Waltz 91]. Every ten years the Census Bureau mails questionnaires to every US household, and uses the collected data (on family members' names, ages, occupations, etc.) to decide how many representatives will come from each state, how government funds will be split between states, etc. Ten percent of households receive the Census long form, which includes many more questions, including ones on the occupations, industries, and job responsibilities of the respondents.

In this project, we developed an MBR (Memory-Based Reasoning -- explained below) system for automatically classifying Census long form respondents into one of 232 industry categories and one of 504 occupation categories. Twenty-eight million long forms had to processed in 1990. In 1980, all long forms were classified by hand. In 1990, the Census Bureau used AIOCS, a rule-based expert system built during the 1980's to classify long forms [Appel & Hellerman 83; Appel & Scopp 87]. Tests showed that AIOCS was able to classify about 47% of the returns with an accuracy greater than or equal to human classifiers -- the remaining 53% had to be classified by hand. Our system, which uses the Memory-Based Reasoning (MBR) model, was run on the same data used to test AIOCS, and substantially outperformed it, processing about 61% of the returns at the required accuracy. Given that it would have cost about $15 million to hand-classify all 28 million forms, the MBR system would have saved over $2 million in hand-coding costs compared to the expert system. (The Connection Machine® system that the MBR system ran on cost about $1 million.)

But the most important characteristic of the MBR system is the speed and cost with which it was engineered and constructed. The MBR system required only four person-months to build, compared with 192 person months for the expert system, a savings of 98%! The reason is that the MBR system was able to use as a key component the database of 132,247 cases that the Census Bureau had constructed to test its expert system. (The construction of this database was not included in the 192 person-month project time for the expert system.)

Results are summarized in table 1, below:

	% of database handled		software effort in person-months
	industry codes @ 11% error rate	occupation codes @ 14% error rate	
MBR	67	53	4
AIOCS	57	37	192

Table 1

3. The Memory-Based Reasoning Paradigm

MBR (Memory-Based Reasoning) methods use large databases of actual phenomena to automatically build systems to handle a very broad range of phenomena [Stanfill & Waltz 86]. In MBR, each new example to be classified is compared with EVERY previous example, and the best match (or in one variant the k nearest matches) is used as a precedent to show how to handle the new example. In order for this method to work well, the database used for comparison must be large enough to contain most of the phenomena ever seen, and large enough so that rare phenomena appear with approximately their true frequency.

Although the idea of MBR is conceptually simple, methods for matching cases are not always obvious. For example, in the Census database, most of the fields are unconstrained free text. For this task, we used methods borrowed from IR (Information Retrieval); text fields are compared using a weighted overlap metric, where the 'score' of a match is the sum of the weights for each of the words or terms (e.g. pairs of words) the two fields have in common. Weights can be chosen according to their information value (computed according to their frequency in the overall database). Alternatively, weights can be chosen on a per-category basis, that is according to how well correlated each word or term is with each particular category. See [Creecy, Masand, Smith & Waltz 91] for details.

An MBR system consists of two main components, a 'shell' and a database of classified cases. The shell contains the user interface, the database handling tools, the similarity metrics, and the mechanisms for combining information and choosing the best precedent(s). The shell is relatively small, and can be largely reused to construct new applications. The database required is similar in form to the 'training sets' constructed for training and testing artificial neural nets.

MBR has a number of advantages over expert systems, neural nets [Rumelhart & McClelland 86; Waltz & Feldman 88], and decision-tree building systems [Quinlan 83]: 1) it is easy to implement -- most of the effort in writing the MBR system goes into devising and testing metrics for judging the degree of similarity between examples; 2) it is easy to update -- new items can be added, deleted, or modified at any time, and the results of the modification are used the next time a decision is made; 3) the system can justify its decisions or actions by giving the precedent that

it used; and 4) the system can estimate its confidence in its actions -- if a new example exactly matches a previously encountered example, it can handle the new example with confidence, while if no precedent matches closely, the system can say that it has little confidence in the appropriateness of the closest precedent. Even the strangest and rarest examples in the database are available for classifying new returns, whereas, in an expert system, rules for such cases are very unlikely to be written, and in a neural net system, unlikely to be learned. The only drawbacks of MBR are 1) it requires hardware with large amounts of memory and computing power, preferably a massively parallel machine, on which the MBR model fits very naturally, such as the Connection Machine system (this is a classic case of trading off compute cost for system performance and ease of building a system); and 2) MBR systems will generally not make decisions as rapidly as do neural nets, even though the systems they must operate on are far more powerful and more costly.

4. Why does this work? Why is this a good match for applications?

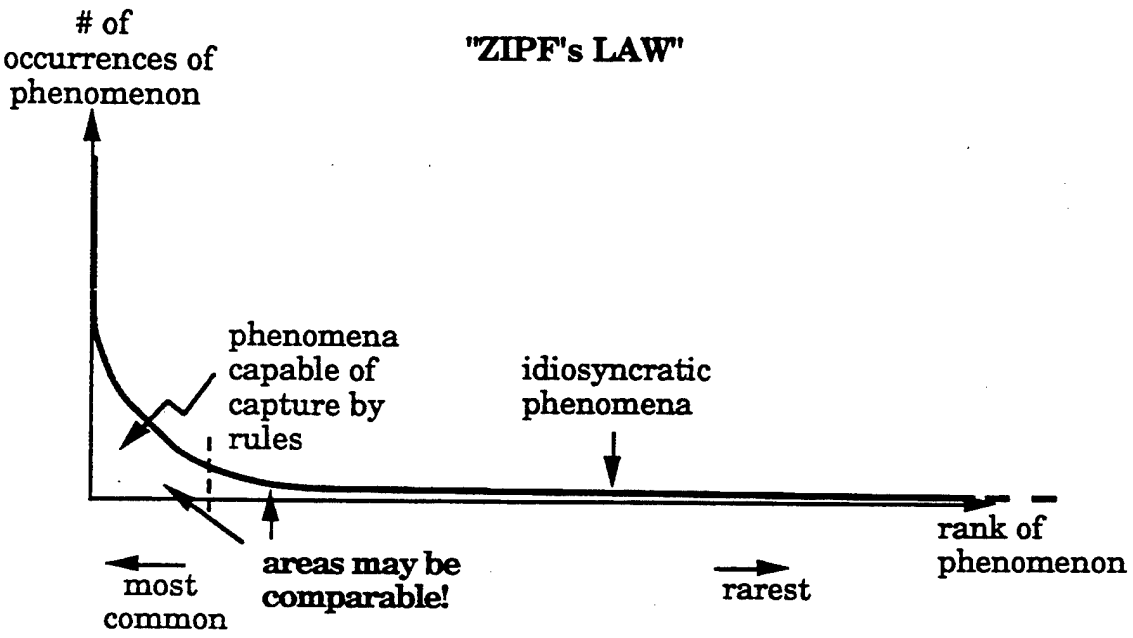


Figure 1

In many domains, phenomena have a characteristic distribution, commonly referred to as Zipf's law: (see figure 1). [Zipf's law states that the frequency of a phenomenon is proportional to 1/rank, where the rank of a phenomenon is a number that represents its position in the list of all phenomena, ordered from most common to rarest. Note that this distribution falls off rapidly, but has a very long tail.] This "law" was originally devised to show the distribution of word frequencies in text, but the same curve applies to many very different natural phenomena, for example pronunciations of letters in English words, occurrences of syntactic constructions in English, failure rates for electrical, mechanical, or software systems, occurrences of diseases, sizes of cities, etc., etc. The most common phenomena occur often, and therefore their regularities are quite striking. One can formulate rules that capture these regularities, and these rules will apply to a substantial fraction of all phenomena encountered. One might

believe that by doubling or tripling the number of rules, one could capture virtually all phenomena. Alas, this belief is unfounded: as phenomena become rarer, their frequency of occurrence becomes very low. However, and this is important, the total number of TYPES of phenomena may be large enough that the total number of rules required to capture all phenomena is on the same order as the total number of phenomena!

When only small memory, serial computers were available, rules (e.g. expert systems and statistical pattern recognizers) were the only feasible method for categorizing, recognizing, or modeling phenomena. Such systems are brittle (i.e. they exhibit hard failures when examples differ even slightly from ones used to test the system), generally disappointing in their coverage (except in the simplest domains), and difficult to build (generally an expert must work with a knowledge engineer to construct the set of rules -- a time consuming and expensive process).

	MBR	Artificial Neural Nets	Rule-based Expert Systems	ID3 Decision Trees
Ease/cost to implement	+	+	-	+
Justification provided?	+ (precedent)	- (no)	+ (chain of rules)	? (possible)
scales to large domains?	+	+	-	+
Allows mixed data (#'s, text...)	+	-	+	-
Handles difficult cases	+	+	-	+
Noise tolerant?	+	+	-	-
Easy to Update?	+	-	-	-?
Computationally cheap?	-	+	+	+

Table 2

With MBR, however, as long as there is a database available of examples coupled with classifications, actions, meanings, etc., an application can be developed readily, and the application is not limited to coverage of only the common, patterned phenomena. Soft or fuzzy match metrics have proven fairly easy to devise, and text databases (or mixed text and numerical databases, such as the Census database) can be dealt with readily through the use of IR metrics used in relevance feedback systems [Stanfill & Kahle 86; Salton 71]. Table 2 shows a summary of the relative advantages of MBR compared to expert systems, artificial neural nets, and automated decision tree building systems.

5. Summary

Massively parallel computers allow developers to trade computing power and memory in order to build several kinds of applications with very little human effort. An important benefit of such applications is the improved accuracy and coverage that can result when the domain/database for the application follows Zipf's law. Application domains that obey Zipf's law include text databases; medical diagnosis; troubleshooting; optical character recognition (OCR); robot arm control; and automatic keyword assignment. In addition, other database-related tasks, including text retrieval and marketing applications, are also excellent candidates for massively parallel applications (See [Waltz 90] for more details of work in these areas. In general, large databases have grown much faster than mainframes' ability to handle them; massively parallel machines offer the promise of better quality of performance, much faster response time (in some cases reducing multi-day task times to hours or less, and making what were previously hour-long runs interactive), and dramatically more rapid development of applications. Finally, these tools, especially the database handling tools, can form the basis of highly intelligent applications of the future, as foreseen by the researchers of the Fifth Generation project [Kurozumi 88].

References

Appel, M.V. and Hellerman, E. Census Bureau Experiments with Automated Industry and Occupation coding. *Proc. Amer. Statistical Assoc*, (1983), pp. 32-40.

Appel, M. and Scopp, T. Automated Industry and Occupation Coding. presented at Development of Statistical Expert Systems (DOSES), December 1987, Luxembourg.

Brooks, F. No Silver Bullet. *IEEE Computer*. vol. 20:4, (April 1987), pp. 10-19.

Brooks, F. *The Mythical Man-Month*. Addison-Wesley, Reading, Mass., New York, 1975, Chapter 14.

Creecy, R., Masand, B., Smith, S. & Waltz, D. Trading MIPS and Memory for Knowledge Engineering: Automatic Classification of Census Returns on a Massively Parallel Supercomputer. Submitted for publication, 1991.

Kurozumi, T. Present Status and Plans for Research and Development. *Proc. Int'l. Conf. on Fifth Generation Comp. Sys*, Vol 1, (Nov. 28-Dec 2, 1988), pp. 3-15.

Quinlan, R. Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. Carbonell, and T. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing, Los Angeles, CA, (1983), pp. 463-482.

Rumelhart, C. and McClelland, J., et al. *Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1986.

Salton g. *The Smart Retrieval System -- Experiment in Automatic Document Classification*. MIT Press, Cambridge, MA, 1971

Stanfill, C. and Kahle, B. Parallel free-text search on the Connection Machine System. *Comm. ACM 29* 12 (December 1986), pp. 1229-1239.

Stanfill, C. and Waltz, D. L. Toward Memory-Based Reasoning. *Comm. ACM 29* 12 (December 1986), pp. 1213-1228.

Waltz, D. L. Massively Parallel AI. *Proc. National Conf. AI,* (AAAI '90), Boston, (August 1990).

Waltz, D. & Feldman, J. *Connectionist Models and their Implications*. Ablex, Norwood, NJ, 1988.